(2)

AN INQUIRY INTO THE COSTS OF POST DEPLOYMENT SOFTWARE SUPPORT (PDSS)

THESIS

Mark A. Collette
Captain, USAF

AFIT/GLM/LSY/89S-10

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY
AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

2

AFIT/GLM/LSY/89S-10

AN INQUIRY INTO THE COSTS OF POST

DEPLOYMENT SOFTWARE SUPPORT (PDSS)

THESIS

Mark A. Collette
Captain, USAF

AFIT/GLM/LSY/89S-10

The contents of the document are technically accurate, and no sensitive items, detrimental ideas, or deleterious information is contained therein. Furthermore, the views expressed in the document are those of the author and do not necessarily reflect the views of the School of Systems and Logistics, the Air University, the United States Air Force, or the Department of Defense.

AFIT/GLM/LSY/89S-10

AN INQUIRY INTO THE COSTS OF POST DEPLOYMENT SOFTWARE SUPPORT (PDSS)

THESIS

Presented to the Faculty of the School of Systems and Logistics of the Air Force Institute of Technology

Air University

In Partial Fulfillment of the Requirements for the Degree of Master of Science in Logistics Management

Mark A. Collette

Captain, USAF

September 1989

Preface

This research was the result of the concern expressed by many people on the lack of accurate information available for support cost estimation models. The purpose of this research was to determine what information was required, and from this information, suggest improvements to increase the accuracy of the collected information. The results of this study should be useful to those who work with software support on a daily basis, and to those interested in support cost estimation.

This thesis would not have been possible without the unselfish support of many people. I would first like to express my appreciation to my thesis advisor, Professor Daniel V. Ferens, for his guidance, encouragement, and support in completing this research. I am also grateful for the support I received from the ALCs and HQ AFLC throughout this thesis process. The guidance given by Major Mathew E. Cvitanovich, from the Air Force Cost Center, was a tremendous help in shaping the overall direction of this thesis. Finally, I would like to thank my wife Bow for her understanding, support, and love throughout this long endeavor.

Mark A. Collette

Table of Contents

List of Figures

List of Tables

AFIT/GLM/LSY/89S-10

Abstract

The increasing cost of software maintenance is taking a larger share of the military budget each year. As new weapon systems are acquired, these support costs must be accurately estimated and budgeted for in the total cost of acquisition. In particular, the accurate tracking of Post Deployment Software Support (PDSS) costs is critical to the overall estimation process, yet it is one of the most difficult areas to document.

This study had three objectives achieve to be considered successful:

(1) Determine whv PDSS is so costly and difficult to estimate.

(2) Determine what factors make PDSS costs fluctuate.

(3) Develop possible changes to the PDSS tracking system to improve availability of information.

The conclusions that were reached by this thesis indicate that the redesign of the AFLC Form 75 is a partial solution to the software support tracking system. The use of the Form 75 as a funding document would appear to be a necessary step in the process toward a more complete cost estimating system.

AN INQUIRY INTO THE COSTS OF POST DEPLOYMENT SOFTWARE SUPPORT (PDSS)

I. Introduction

General Issue

One of the most important budgetary issues currently being discussed is cost determination for weapon system acquisitions. With cost overruns measured in millions of dollars, an accurate budgetary estimation of the total cost to acquire and maintain a weapon system has become essential. This budget is necessary to allow for the economical purchase, operation, and maintenance of military weapon systems. An area that has a large effect on the cost of weapon systems is computer software support. In fact, the rising costs associated with maintaining and supporting computer software are overriding the cost of acquiring the original computer hardware and software (6:18).

Problem Statement

The accurate tracking of Post Deployment Software Support (PDSS) costs is critical to the overall cost estimation process, yet it is one of the most difficult areas to document. Currently maintenance support of Embedded Computer Systems (ECS) software is estimated in Air

Force Logistics Command (AFLC) by the use of the Computer Program Configuration Sub-Board Item Record, AFLC Form 75. Unfortunately there is no procedure to relate actual costs to the estimated costs. This severely restricts the use of software support cost models such as the Constructive Cost Model (COCOMO) and the Avionics Software Support Cost Model (ASSCM).

The reason this tracking of costs is so important is that software maintenance managers are beginning to feel the impact of having to support an ever increasing number and complexity of programs. In the F-111 approximately 10,000 source lines of code were used, while the B-1B uses approximately 1,200,000 source lines of code (8:46). This increase in complexity coupled with the fact that the life span of software is increasing, through reusable code, has caused the software maintenance phase to become a significant area of concern for managers and programmers alike.

There are many factors that are causing difficulty in this growing field of the software life cycle called maintenance. Part of this problem can be traced back to the lack of life cycle perspective. In other words, managers are being forced to produce on schedule and under budget. This means that the downstream costs (maintenance) are ignored, and are absorbed later as overrun or modification costs in the PDSS phase of the system life cycle (14:17).

In spite of the efforts that have been devoted to improving the reliability and maintainability of all weapon systems under the program R&M 2000, reliability and maintainability of software are not placed on the same level of importance as cost, schedule and performance.

Research Objectives

During the course of this study the following steps will be accomplished:

(1) Determine why PDSS is so costly and difficult to estimate.

(2) Determine what factors make PDSS costs fluctuate.

(3) Develop possible changes to the PDSS tracking system to improve availability of information.

The hypothesis for this study is to determine if it is possible to develop a PDSS cost tracking system that would be easy to use and provide accurate and timely information.

Scope and Limitations

Scope. This study will focus on mission critical computer resources (MCCR) software support for Air Force weapon systems. Software support will be defined as in AFR 800-14, Lifecycle Management of Computer Resources in Systems, which states:

> The sum of all activities required to ensure that, during the Deployment phase of a computer system's life cycle, the implemented and fielded software fulfills its original mission, any subsequent modifications to that mission, and any requirements for product improvement. (12:22)

Limitations. This study will necessarily be limited due to time constraints and funding limitations. This study will also be limited by the amount of information available from HQ AFLC, HQ AFSC, and the associated Air Logistics Centers.

II. Background

This section of the study will describe the background and present a literature review on software support and maintenance. One of the first areas to be addressed will be to provide working definitions as a basis for exchange of ideas. The first area to be discussed will be the efforts involving the software support estimation model ASSCM and the software cost model COCOMO. Following the models, the next areas of the literature review will be devoted to the discussions involving why PDSS is expensive, the determining factors behind PDSS, and what metrics in PDSS must be tracked.

Definitions

Software Support. One of the most confusing areas of this study was the wide variety of terms used for the same item. The greatest confusion involved software support, also known as software maintenance, sustaining engineering, redevelopment, software renovation, and software evolution. (20:16) No matter what it is called, most people agree that it is divided into the three categories of corrective, adaptive, and perfective actions. (21:495) One of the biggest misconceptions about maintenance is the percentage of problems in each area. Recent studies have shown that perfective, adaptive, and corrective maintenance account for 60%, 20%, and 20% of the effort respectively (19:9-10).

ASSCM

In 1979, the Air Force Wright Aeronautical Laboratories (AFWAL) contracted for a study conducted by Hughes Aircraft Company to predict PDSS costs for avionics embedded computer systems. The study was designed to gather information about producing a model that would predict total software support costs early in development. This model would allow the proper software design alternatives to be chosen, thus obtaining required performance at the optimal life cycle cost (23:V). In 1980, after this first study was completed, the contract award to write the prediction cost model, was given to Systems Consultants, Inc. (SYSCON) in Washington, D.C., and the Avionics Software Support Cost Model (ASSCM) was completed in 1982. The OFP software studied basically controls navigation, weapon delivery, and fire control. In the SYSCON study the OFP software data for the baseline was obtained from the F-111F, F-16, and the A-7. This data was validated in the model against the FB-111A (22:16). One of the difficulties in the original study was a lack of data. To compensate for this, a Delphi technique was used to obtain the experts' point of view for how long each phase of the software support should take, and to allow the people a chance to achieve consensus. This technique is often used when in-depth thought is required to solve a problem; it allows each person to express their own views and also benefit from other opinions (9:465-6).

Model Overview. The Original ASSCM is a predictive model that performs estimations of total life cycle support costs for embedded computer software. It was designed to handle Operational Flight Programs (OFP), Electronic Warfare (EW), and Communications Electronics (CE) applications. The model projects cost estimates, in 1981 dollars, for systems whose expected life is between 1970 and 2025 (22:26).

Underlying Assumptions. There were several assumptions that were made when SYSCON developed the ASSCM, including:

(1) Each Air Logistics Center (ALC) collects data costs independently and differently.

(2) Too few weapon systems were available to develop significant statistical cause-and-effect relationships.

(3) Labor cost data was projected on Form 75 for all ALC's, but there was no record of the actual labor.

(4) Indirect labor costs, and support equipment costs are not broken out by system.

(5) There are certain system characteristics that have a direct effect on software support costs, including program size, language, complexity, structure, and others.

(6) An algorithm that can be used with limited available data must be developed for an accurate cost estimation model in this situation, to best achieve the needs of the Air Force (22:7-8).

Context for Model Use. The ASSCM can be used by the weapon system program manager to estimate total life cycle support costs. These costs will encourage the manager to make intelligent choices in software and hardware development. Certain choices will be justifiable when the overall life cycle support costs are examined.

COCOMO

The COnstructive COst MOdel (COCOMO), described in this thesis, will be focused on its usage toward estimating software maintenance costs. Software maintenance, as described by Barry Boehm, is separated into either software updating or software repair. The main difference between these two categories is that in the software updating the actual functional specification is changed, while in software repair it is left intact. Software repair is then categorized into corrective, adaptive and perfective actions (6:536).

Model Overview. The development of COCOMO was based on the work that was done by Dr. Barry W. Boehm while he was working at TRW. In Boehm's own words:

> The initial version of COCOMO was based on a review of existing cost models, a two-round Delphi exercise involving 10 experienced software managers in estimating effort for various cost driver attributes, and experience with several software cost-estimation models at TRW. The initial COCOMO model, which had only a single development mode, was calibrated using 12 completed projects. The resulting model was then evaluated with respect to a fairly uniform sample of 36 projects, primarily aerospace applications, producing fairly good agreement between estimates and actuals. (6:492-3)

COCOMO was developed through the use of a data base with 63 software projects that had a wide representation of "real-world" data. Included were 23 organic, 12 semidetached and 28 embedded data points along with 7 business, 10 control, 13 human-machine, 17 scientific, 8 support and 8 systems types (6:83).

There are three primary levels of COCOMO; Basic, Intermediate, and Detailed; with each version having three modes: organic, semi-detached, and embedded. This combination of levels and modes can be thought of as a matrix that becomes increasingly detailed and, consequently, more accurate. The Basic COCOMO is used to make rough estimates and is limited due to the use of only one cost factor, thousands of delivered source instructions (KDSI), as shown in Table 1. In this case the item being determined is man-months (MM) of effort to develop a complete project. This ignores the influence of other factors such as hardware constraints, ability of personnel, programming techniques and software limitations (6:58).

Table 1. Basic COCOMO Effort Equations (6:75)

Mode	Effort
Organic	$MM_{DEV} = 2.4(KDSI)^{1.05}$
Semidetached	$MM_{DEV} = 3.0(KDSI)^{1.12}$
Embedded	$MM_{DEV} = 3.6(KDSI)^{1.20}$

A stand alone version of the Basic COCOMO is used for software maintenance estimation, which is essential to determine life cycle cost. Trying to plan for the total life cycle of anything, much less software, is understandable but quite difficult to accomplish. This effort for maintenance has been estimated, by federal managers responsible for software applications, to consume approximately 60 to 70 percent of the resources available for application software (18:2). Many attempts have been made to try to estimate these costs and this effort is still under way. One of the methods that has been tried is the modification of the Basic COCOMO to better reflect the effort required for maintenance.

The COCOMO maintenance model is based on two fundamental assumptions. The first is that maintenance costs are driven by the same types of factors that influence software development costs (6:536). This is based on the definition of activities included in software maintenance to include:

> Redesign and redevelopment of smaller portions (less than 50% new code) of an existing software product
>
> Design and development of smaller interfacing software packages which require some redesign (of less than 20%) of the existing software product
>
> Modification of the software product's code, documentation, or data base structure (6:54)

The second assumption is that it is possible to predict the percentage of code that will undergo change in a typical year. This Annual Change Traffic (ACT) is used in conjunction with the KDSI in the Basic model to determine annual man-months of maintenance [$(MM)_{AM}$] effort (6:536).

TABLE 2. Basic Annual Maintenance Effort Equations (6:536)

Mode	Effort
Organic	$MM_{AM} = 2.4(ACT)(KDSI)^{1.05}$
Semidetached	$MM_{AM} = 3.0(ACT)(KDSI)^{1.12}$
Embedded	$MM_{AM} = 3.6(ACT)(KDSI)^{1.20}$

When a more accurate estimate of the cost involved with a software project is required, more factors that influence cost may be included. The Intermediate COCOMO uses 15 factors in addition to the cost factor used in the Basic model to more accurately determine cost. These additional factors more accurately account for the influence of software product, computer, personnel and project attributes. These factors are then used as multipliers to adjust the nominal effort estimate, similar to the equations used in the Basic COCOMO. These estimates are useful in determining overall effort required for completion, and they also help in sensitivity analysis and in-depth comprehension of the total software project (6:114-7).

The Detailed COCOMO tries to compensate for two main deficiencies in the Intermediate COCOMO: errors in estimating amount of effort by phase and the unwieldy effort required in a large project with many components. Correction for errors in estimating effort by phase is attempted through the use of phase-sensitive effort multipliers. These multipliers consider the differing amount of effort that is affected by each attribute in each phase of development. The variety of components in a large project are grouped by system, subsystem and module level to reduce the number of calculations required. This grouping is referred to as three-level product hierarchy. Another capability in the Detailed COCOMO is the ability to adjust the phase distribution of the development schedule (6:344-8).

As shown in Fig. 2-1, as a person moves down the matrix, the freedom available for programming decreases. In the organic mode each program is separate and requires little communication or interface with other programs. This mode also has a fairly stable development environment, with minimal changes in hardware or operational procedures and the personnel are familiar with this programming or have worked on similar programs. This type of program has little need for exploring new techniques or algorithms and has a relatively small size. [In this instance, 50 KDSI (Thousands [K] of Delivered Source Instructions), or less is considered to be small size.] They also receive little incentive for early completion (6:78-9).

	BASIC	INTER-MEDIATE	DETAILED
ORGANIC	Simple Programs/1 Cost Driver	15 Additional Cost Drivers	Phase-Sensitive Effort Multi-pliers and
SEMI-DETACHED	Some Limits		Three-Level Product Hier-archy
EMBEDDED	Constrained Critical Compliance		

Figure 2-1. Level/Mode Matrix

The semi-detached mode can be thought of as a combination of organic and embedded characteristics. An example of this would be working with an assortment of programmers, some of whom are experienced in the system and some who are brand new. Also, there may be some flexible interfaces with other programs and some rigid requirements for other interfaces. This mode is usually found in programs with up to 300 KDSI and has some premium on early completion (6:79).

The embedded mode is designated for programming projects that must operate within tight constraints, in a changing environment, with programmers exploring new algorithms and designs. These factors lead to a high premium on the early completion of projects to prevent making additional changes and early obsolescence. These types of projects come in all sizes; and the high cost of changing other interrelated systems reduces the number of options in flexibility in the embedded system interfaces (6:79-80).

Underlying Assumptions. There are are several assumptions that must be made to effectively use COCOMO in any mode or at any level. The first assumption that must be made is that it is possible to estimate KDSI early in the software life-cycle. Second, the KDSI must be the primary cost driver in the project. Third, only the direct labor involved in the project is estimated. In the direct labor area, approximately 152 hours of working time per month is used. In addition, it is assumed that good management procedures will be used throughout the project by both the developer and the eventual user. This would indicate that there would not be significant modifications or changes in requirements specifications after the planning and requirements phase. COCOMO is primarily designed for the development of software and therefore is used to estimate the costs involved from the start of product design to the end of the test phase. When other phases of the software life-cycle need to be estimated, that can be done, but must be done separately (6:58-9).

Context for Model Use. In most applications these assumptions can be readily accepted, but there are exceptions. Some researchers have suggested that the suggested 152 hours of productivity in a typical month is way out of line. They suggest that, out of a standard of 172 hours per month, a typical programmer will be available approximately 86 hours per month (7:89-90). In relation to Air Force software maintenance, a more realistic estimate, according to the most recent Air Force Management Engineering Agency study, would be an average availability of 145.1 hours per month (4:12).

III. Research Objectives

PDSS

An integral part of this study will be the identification of the causes for the wide variation of costs associated with Post Deployment Software Support (PDSS). The PDSS can include different varieties of maintenance such as corrective (fixing a problem), adaptive (making changes to design or function) or perfective (enhancing performance) maintenance (6:534-6). These changes are not affected greatly by the cost of materials or supplies but rather by the personnel costs. To paraphrase Lord Webb-Johnson (25:16), there are three type of personnel involved in software support:

> The design programmer who builds a castle in the air
> The maintenance programmer who lives in it.
> And the software manager who collects rent on it.

Determining Factors. Two of the factors that influence the costs associated with these changes include the skill level of the personnel involved with the original design and the skill level of the personnel making the changes. The original design personnel can make the support actions of maintenance personnel difficult by not using logical programming steps and by not properly documenting the process involved in creating the original program. The skill of the maintenance personnel is difficult to measure because programming is a creative process that cannot be exactly quantified. The program that one programmer can develop in ten days and 10,000 lines of code can take another programmer ten hours or ten weeks and 1,000 or 20,000 lines of code; and both programs may perform the same function. This becomes an even more difficult item to estimate when one programmer has to think like another programmer to modify a previously written program in a limited amount of space (22:70-72).

One of the major sources of data for this study will be the data call conducted by the DoD in 1988 to analyze software maintenance costs. The estimates provided by Headquarters USAF come complete with its own disclaimer that states:

> We have attached cost information for maintenance of software in embedded computers to support your studies. Be advised that all the resource estimates are just that -- best estimates of requirements at the time the information was collected. (5:1)

Expense. Use of a disclaimer is understandable when the numbers are examined. According to the summary of the data call, the Air Force paid over $68 million in organic software maintenance costs in FY87 and over $92 million in FY88. This would mean an increase in cost of more than 35 percent per year. The predicted costs for FY89 for organic support is only a little over $106 million, an increase of only 15 percent that year.

Metrics. If software is thought of as an abstraction as compared to hardware which is tangible, then software maintenance is even more abstract as compared to software development. This is because software does not wear out like hardware does. Hardware is subject to deterioration (wear out) in the course of normal use and requires maintenance to restore it to its former operating state. Conversely, software does not change unless someone changes it. Software maintenance does not mean restoring the software to its former operating state; but, instead, means making changes away from its former state. In hardware, the former state was the ideal and deterioration has caused degraded performance. The restoration of hardware to the original operating condition will restore optimal performance. The problem with software, however, is that defects or deficiencies in the former state will have caused the degraded performance, and software must be changed from the original in order to obtain optimal performance. Software maintenance becomes a process in which the software is continually changed in order that its performance ma be improved, or at least maintained. This maintenance is often so poorly done that the software's performance is neither sustained nor improved.

IV. Current Tracking and Estimating Systems

The management of software maintenance requires that the individual pieces and versions of software be identified. This identification of software is required because one data tape or disk is identical to another unless a line by line comparison of the code is done. Inherent in this estimating and tracking system should be the ability to monitor the amount of hours and dollars spent on supporting the systems. The Air Force currently has several methods in place to accomplish this task including the use of Computer Program Identification Numbers (CPINs), Element of Expense Investment Codes (EEICs), AFLC Form 18 and AFLC Form 75.

CPIN

The CPIN system, established in the early 1970s, was one of the first Air Force systems designed to uniquely identify computer software. This centralized data system was automated in the early 1980s and contains five basic functions including CPIN assignment, compendiums, establishing requirements, distribution, and management reports (13:6-9).

One of the most important functions of the CPIN system is the assignment of CPINs to provide a means to track computer software. The CPIN is not required while the software is still under development, but is required once the software is delivered to AFLC (3:6). It is mandatory that the CPIN be referenced in all technical communications, and recommended to be used in all managerial communications. The CPIN assignment is only possible once an AF Form 1243/1244 is submitted to the Oklahoma City ALC, though emergency CPINs can be assigned by phone on a limited basis (13:6-9).

A compendium, as defined in Webster's is "a summary or abstract containing the essential information in a brief form" (26:298). The CPIN compendiums provide essential information of computer software configuration items (CSCIs) and engineering documentation to users with known requirements at the CPIN office (13:6-9).

The establishment of requirements for CSCIs and compendiums is a process that utilizes the G022 automated T.O. system. When this material is needed, the requesting agency forwards a completed AFTO Form 157 to their T.O. Distribution Office (TODO), who then forwards the request to the appropriate agency. Even though the Oklahoma City ALC manages the overall CPIN system and publishes compendiums, the CSCIs are distributed by the managing ALC Software Control Center (SCC) (13:6-10).

The distribution of a compendium or CSCI also relies upon the G022 T.O. System. When this material is ready for distribution, mailing labels are generated based on requirements stored in the T.O. system. The managing ALC SCC distributes the required material to the requesting

TODO, who then forwards the information to the appropriate subaccount (13:6-10).

Provisions are also made in the CPIN system for the automatic generation of management reports that list such things as who is receiving material and how many copies they are receiving (13:6-10). These reports help keep track of who is utilizing various versions of software and overall trends in software in the Air Force.

EEIC

The tracking of costs involved with various steps in the software life cycle are especially relevant to the support function. In the Department of Defense (DoD), the Six Year Defense Program (SYDP) serves as the financial data base for all DoD activities (1:13-5). This means that the SYDP can be used to help develop budget estimates that are based on approved forces and missions. Most money for the maintenance of software is received through Operation and Maintenance (O&M) Appropriations.

> This appropriation (O&M) provides for financing the day-to-day operating and maintenance costs of Air Force activities. These funds include, but are not limited to, monies for sustaining engineering programs, contract services for maintenance of equipment and facilities, fuel, supplies, and other weapons systems support activities. (2:3.3)

O&M is just one of the categories of appropriations that is an input to the Six Year Defense Program (SYDP) and then broken out into smaller subgroups that are called major force programs (MFP). These MFPs include such areas as

Strategic Forces, General Purpose Forces, Airlift Forces, and General Supply and Maintenance (1:13-14).

This O&M money is then distributed to the various bases on an annual basis. This money is further distributed to the various Responsibility Center (RC) managers who then distribute the money to the individual Cost Center (CC) managers. The CC manager will provide an estimated operating budget each year to the RC manager for consideration in the following yearly budget. The CC manager justifies his yearly estimated operating budget by demonstrating the trends and totals that have occurred in the previous year, in his various Element of Expense/ Investment Codes (EEICs). These EEICs are used to identify individual expense activities such as TDY Transportation Expense, TDY Per Diem, Service Engineering by Contract, and other Miscellaneous Expenses (2:3.3).

This method of funding should serve two main purposes: the tracking of past expenses and the estimation of future expenses. With differentiated programs and more specialized EEICs, the opportunity would exist in the ALCs to determine exactly how much was spent to support each phase of software maintenance for any given weapon system. One of the problems that exists is that there is very little analysis that is done on the amount of money that is spent on software maintenance. This was made perfectly clear when the latest report on software maintenance in DoD stated:

> OSD is cognizant of plans, policies, and procedures for the maintenance of hardware systems. However, visibility in these areas has not been established for the maintenance of software. To integrate hardware and software maintenance requirements for weapon systems, OSD needs software maintenance program data. (11:A-2)

Current AFLC Form 75

Whenever anyone is considering making changes to software, it is important to maintain control over what is proposed and what is actually done. This is why it is important to have a formal written procedure and standard form for any changes that are under consideration. The acceptance of a requested change in software is normally processed through the Computer Program Configuration Sub-Board (CPCSB). This board typically consists of division chiefs from the Materiel Management (MM) directorate, Maintenance (MA) directorate, and other ALC directorates as needed (3:13). Currently the form that is used by AFLC for this purpose is the AFLC Form 75 (see Appendix A). This form, also known as the Computer Program Configuration Sub-Board Item Record, performs a variety of functions including administrative, technical, and managerial requirements. The Form 75 is prepared as the result of a discrepancy report completed by either the user or maintainer in response to discovered errors, changing needs, or new requirements.

The first few entries on the Form 75, (see Appendix B for more details) are administrative, with the identifying of the date the form was prepared, the system designator, and the weapon system or ground system being supported (3:59). The estimated severity of the change is then given, to help the members of the CPCSB understand the severity of the change. Background information is then given as to who is requesting the change, whether it is a software only change, what kind of equipment it is on, whether the job is to be done in house, and whether or not this is a update of previous software (3:59).

The CPCSB is then shown who will be involved with the change, identifiers for the change, the priority of the change, and related changes that are being developed or affected by the change. The next section provides a verbal description of the purpose of the change, how the deficiency will be resolved, and how long each step will take.

The last item, on the front of the Form 75, is a graphical timeline for the completion of the software change. One nice feature about this timeline, is the ability to label the increments as needed (3:59).

The second page of the Form 75 starts with a section stating how the hours, and consequently the funds, will be allocated to solve the problem. This is categorized by in-house versus contract labor. In conjunction with the funding section, a checklist is available to determine if the change was budgeted for, and whether the hardware modification was funded. An additional question is asking whether the proper people have been notified of the change (3:60).

To make the CPCSB aware of the importance of the software change, an impact statement is provided as to what the implications are if the change is not accepted. This impact statement focuses on the possible degradation of the mission capability if the change is not implemented. Also in this area, any additional necessary comments are included (3:60).

The AFLC Form 75 is useful for general information, but it has many shortcomings that will be discussed in the Research Summary in the next chapter. Now, more than ever, the costs associated with maintenance, especially in software, must be fully recognized. To achieve this recognition, certain changes to the Form 75 should be considered. These suggestions for change are presented in the Recommendations section of Chapter V.

V. Conclusions and Recommendations

The purpose of this research effort was to shed some light onto the murky world of Post Deployment Software Support (PDSS). To accomplish this task, background information was provided to enable everyone to obtain a basic understanding of what models have been developed to work with PDSS, what PDSS is, and how PDSS is currently being tracked. This final chapter will describe a possible alternative to the current method of PDSS tracking, along with other conclusions from the research.

Research Summary

Even though this was a very limited study, certain ideas became evident. The first idea that appeared, demonstrated that it is very difficult to develop a solution to a problem, when you can not agree as to what the problem is. This was shown when the discussion was presented on ASSCM and COCOMO.

ASSCM. The ASSCM development study, performed in 1980-81, stated that the ALCs were not using consistent procedures for the collection of software support information (22:7). Following this development study, SYSCON developed a model based on the opinions of experts in the field (Delphi Technique). One of the questions that occur with a technique like this is, "How accurate is this estimate?". The reason the experts were asked for their opinion in the first place, is that we have no records of what it actually costs to perform software support. Since we do not keep accurate records on software support costs how would these experts know what the actual costs are? It is difficult to imagine one person keeping track of all the software support costs involved with a major software system.

The model would have been much more believable if it was not for the validation procedure that was performed. After the original model was developed, "SYSCON conducted an exhaustive search to complete its data base as best it could. This resulted in complete cost and characteristic information for two systems" (22:83). When these two systems were run through the model, the model miscalculated direct labor costs by -26.6% and +31.2% (22:84).

COCOMO. The work that has been done by Barry Boehm on the COCOMO model has become a semi-standard for the software industry. The information that he has gathered is the basis of one of the most extensive data bases on software that has constructed. In conjunction with his research, Barry Boehm has been a prolific author with articles that have been published by Rand, TRW, Institute of Electrical and Electronics Engineers (IEEE) Transactions on Computers, IEEE Transactions of Software Engineering, numerous reports for the Air Force, and his book, Software Engineering Economics, has become a classic in the field of software.

Even such a world renown expert has trouble with the estimation of software costs, both in development and maintenance. In fact, in his book, Boehm states:

> Today, a software cost estimation model is doing well if it can estimate software development costs within 20% of the actual costs, 70% of the time, and on its own home turf (that is, within the class of projects to which it is calibrated). Currently, the Intermediate and Detailed COCOMO models do approximately this well (within 20% of the actual costs, 68 to 70% of the time). (6:32)

This does not mean that the efforts that have been expended toward software maintenance estimation have been wasted. On the contrary, studies that have been done to understand software maintenance, have produced better software maintainers and developers.

PDSS. Post Deployment Software Support can be improved in many different way including source code, documentation, coding and review techniques, testing standards and procedures, and change control. In reference to source code, James Martin implies that one of the best ways to solve a problem, is to prevent it. In fact, he says:

> The maintainer should participate actively in the program development process by offering maintainability guidelines to the developers, by gathering testing and error information, and by conducting maintainability acceptance audits. Waiting until the operation and maintenance phase to learn about a program, to make support preparations, or to ensure maintainability greatly increases the risk of not being able to effectively and efficiently perform program maintenance. (17:367)

Part of this programmer involvement will have to be more productive programmers. Software professionals are becoming even more limited in supply as the demand for new software increases. It is expected that the software demand will continue increasing at a rate of 25% per year, with the total number of programmers increasing at only 4% per year (16:28). Kitfield then goes on to say:

> If you project current trends in [both commercial and military] software supply and demand out to the year 2040, you find that every man, woman, and child in the country will have to be a software programmer. (16:28)

This brings to mind the question, "if every man, woman, and child is not going to be a programmer, how can we make the programmers we have more productive?" Part of this productivity can be brought about by the use of software programmer metrics. These metrics can include tangible items such as, quantity of work done per day, measured in lines of code (LOC) per day, or even errors found per inspection, measured in errors per 1,000 LOC (20:17-19). These types of measures need to take into account the difficulty of the program and whether a Higher Order Language (HOL) is involved. This indicates that there must be consideration given for quantitative and qualitative measures in the programmer metrics (20:27).

In the maintenance field the importance given to both quantitative and qualitative elements should be extremely high. The maintenance programmer could be measured by the number and complexity of change requests handled, LOC maintained, quality of interface with the user and quality of updated documentation (20:45-49). While these items may not be as easily measured as simply LOC produced per day, they are equally important.

CPINs. The current tracking system for individual CPCIs appears to be in very good shape. Though the CPIN system has existed since the early 1970s, the automation update performed in the 1980s was well done. The CPIN assignment procedure is well documented, the compendiums produced are used by many agencies, the distribution system greatly aids the ALCs, and the automatically generated management reports are relied upon by the ALCs.

EEIC. The premise behind the EEIC funding system is very logical; different categories of money being used for different purposes, however, at the present time the categories are too generic. The software maintenance funding problem is difficult to track and is made more difficult by lumping various costs together. An example of this is the category called software engineering which can use EEIC 54X, 583, 585, 592, and 582 as possible sources of funds (2:4.2). On the other side of the coin, EEIC 54X can be used in a wide variety of activities, such as:

> Used to purchase services from the Depot Maintenance Services, Air Force Industrial Funds (DMS, AFIF). These services include organic and contract (including interservice) depot maintenance for repairing and modifying aircraft and missiles, overhauling engines, overhauling "other major end items", overhauling exchangeable items, supporting area and base tenants, local manufacture and software support. (2:5.8)

As can be seen from this example, if someone wanted to find out how much money was spent on 'Software Engineering', a great many categories would have to be searched. Once the categories containing the information were found, they would have to be literally 'sifted' to find the relevant information. A possible solution to the problem of generic EEICs would be to have a specific EEIC for the category of software maintenance.

Current Form 75. The current AFLC Form 75 is useful in an overall managerial sense, but is limited in the detail required for configuration management. To provide any coherent analysis of cost or effort trends, the Form 75s need to be entered into at least a local data base and preferably an Air Force or DoD wide data base. The administrative areas on the form are good, but the descriptive information in blocks 10 and 14 would be difficult to enter into a data base.

Once the data was entered into a data base it would be possible to show what types of problems are recurring, and suggest alternatives to prevent problems. The data might show that there are problems with certain types or sections of computer programs or even with particular producers of computer software.

The way that the funding is handled is also a problem, because the estimated cost is never reconciled with the actual cost. This lack of verification could lead to the inaccurate estimation of future projects. This would be possible because there is no single source of what the actual costs were on previous projects.

Recommendations

There are many areas that need improvement in the software maintenance arena, and the recommendations that follow will highlight a few points that could be improved. These recommendations are not in order of priority or time sequence and should all be considered equally.

The first possible recommendation is to redesign the AFLC Form 75 to better reflect the information that is required for a cost/effort estimation model such as COCOMO or ASSCM. This is not an endorsement of either of these models, but rather a statement of common sense. The work done by Barry Boehm, in COCOMO, has become a standard in the software industry and should be built upon as much as possible. While the COCOMO model may have to be calibrated for each software system, at least there would a data base available for comparison.

In addition to the information now on the Form 75 the following type of information should be included:

1. Reason for the change such as corrective, adaptive, or perfective maintenance. This information would help the contractors and designers work more efficiently on the next software system.

2. Personnel attributes including the skill level of the personnel available for the change. Depending on the

amount of turnover at the ALC, this could be a semi-fixed number. If the work was to be carried out by a contractor, this number could change for every Form 75.

3. Computer attributes such as amount of space left in memory, or does each bit added mean a bit is subtracted? This would also apply to the amount of execution time that is required for the program. The amount of stability in the overall system is also important, because too many changes in the system means the documentation may not have caught up to reality. While this information may seem excessive, there is no reason why it could not be included as part of the technical orders.

4. Product attributes such as the required reliability of the program and the complexity of the program. Another factor to be added into this is the age of the software and the number of previous changes. The number and frequency of user requirements would be very important in these considerations. This would help indicate the difficulty of working on the program.

5. Project attributes that indicate how the program was written, such as top-down programming and structured programming. Another possible area of concern for the program is whether or not the project is classified. If the project is classified, it will mean additional requirements for TEMPEST approved computer and copying equipment, secure storage areas, and security clearances for the personnel.

6. Number of lines of code in the program and number of lines of code expected to be changed. The current LOC will help illustrate the complexity of the program and is useful in building the data base. The estimated number of LOC to change, should show how well the software engineer understands the effort required for the change. The actual number of LOC changed and final total LOC are also needed.

7. The system language should be included on this form along with the questions, "Was Ada used? If not, why not? Was Atlas used where appropriate? If not, why not?". Since these languages are the DoD standard, they should be used unless there is an overwhelming reason not to use them.

8. The detail on the funding issues should also be clarified, especially with the contract support. Why not show how much time the contractor expects to use for analysis, designing, coding, testing, and documenting. This information is available in the hardware production area, so it could be available for software (10:25).

The physical redesign of the Form 75 is not the complete solution to the software maintenance problem. The procedures that accompany the Form 75 will also have to change. If the Form 75 were used as a funding document, more time and care would be used in accurate estimation of the cost of the changes involved. Using the Form 75 as a funding document would also reduce the tendency to use it as merely a paperwork exercise.

As part of the recommendation to use the Form 75 as a funding document, consideration should also be given to using a charge back system. The charge back system would make the originating user responsible for the cost of the change. Even if no money ever changed hands, this type of system would help provide more information about who is generating the changes and the dollar amount they are responsible for.

In conjunction with the responsibility issue, make contractors responsible for their computer programs. If a warranty program was instituted for software, the ability to maintain the software might improve

These recommendations are possible to implement but would be difficult to enforce. It would take a major shift in the management attitude of "fixing a hardware problem, with a software solution" before any real benefit can be seen from these recommendations.

Problems Encountered

As expected, the biggest problem encountered in this research, was tracking down hard data. While everyone appears to agree that there is a problem with software maintenance, no one has accurate numbers to back them up. The inability to meet face to face with the ALC points of contact, was also a problem but not an insurmountable one. The phone conversations and correspondence with the people at the ALCs, were more than adequate.

Further Research

Further research into the software maintenance question is required and should probably focus on two main areas of data acquisition and trend analysis.

Data Acquisition. Headquarters AFLC is currently developing a new AFLC Form 75 with some of the recommended changes made earlier. Once the new Form 75 is approved, someone needs to collect the information in a centralized location. The individual Form 75s would then have to be deciphered and entered into some kind of data base.

Trend Analysis. Once the database is built trend analysis could be performed on the information available. Some trends to be looked at could include organic vs. contract support for cost and labor hours, comparison of efficiency between ALCs, and overall software trends.

Conclusion

There are many problems with the current state of software maintenance and the recommendations offered here are not presumed to solve all of the current problems associated with software maintenance. However, until the software maintenance problem is better understood there will be no solutions.

Appendix A: AFLC Form 75

COMPUTER PROGRAM CONFIGURATION SUB-BOARD ITEM RECORD

(REFER TO AFLCR [illegible]-21 BEFORE COMPLETING FORM)

DATE

1 SYSTEM AND TITLE

2 PROGRAM/WEAPON SYSTEM BEING SUPPORTED

3 CHANGE
☐ CLASS I ☐ CLASS II

4 ORIGINATOR

5 TYPE OF CHANGE
A. ☐ SOFTWARE ONLY CHANGE ☐ RESULT OF HARDWARE MOP
B ☐ ATD ☐ ATE ☐ C-E ☐ EW ☐ OFP ☐ FSG-70 ☐ OTHER
C. ☐ ORGANIC ☐ CONTRACTOR ☐ OTHER
D. ☐ VERSION CHANGE ☐ REVISION CHANGE ☐ NEW CPCI

6 INVOLVED AGENCIES

7 NUMBERS
CONTROL
CPIN
DR
ECP
MIP
TCTO
CEP
MOD

8 PRIORITY
☐ ROUTINE ☐ SAFETY ☐ URGENT
☐ EMERGENCY ☐ NUCLEAR
☐ BLOCK CHANGE
RELEASE DATE ________

9 RELATED CHANGES (Continue in block 10, if more space is needed)

10 PURPOSE, DESCRIPTION, RELATED ACTIONS/INTERFACES/INTERIM ACTION

11 SCHEDULE

A. PROTOTYPE/DEVELOP												
B TEST/VALID & VERIF												
C DISTRIBUTION (Only)												
D INTERIM ACTION												

AFLC FORM MAR 88 75 PREVIOUS EDITION WILL BE USED

AFLC Form 75 (Page 1 of 2)

12. COST MANHOURS ESTIMATES	MAN HOURS	FUND APPROP/CODE	THOUSANDS OF DOLLARS FY ____	FY ____	FY ____	TOTAL
A. ENGINEERING						
(1) ORGANIC (MAB)						
(2) ORGANIC (MA)						
(3) CONTRACT		EEIC 583				
B. (1) TEST/IMP						
(2) INDEPENDENT VALID/VERIF						
C. DISTRIBUTION						
D. DATA						
E. MATERIALS						
F.						
G.						
H. TCTO/TO						
I. SOFTWARE SUBTOTAL						
J. HARDWARE MODIFICATION						
TOTAL						

13. CHECKLIST

☐ YES ☐ NO COST ESTIMATES ARE IN POM AND ODB FOR REQUIREMENTS IN 12A-G. IF NO, EXPLAIN IN REMARKS.

☐ YES ☐ NO FOR 12H, HARDWARE MODIFICATION IS FUNDED.

☐ YES ☐ NO IF THIS IS AN OFP, EW OR C-E CHANGE, DO ATE AND ATD COMPUTER PROGRAMS NEED TO BE CHANGED IF YES, NOTIFY SM OR IM

14. IMPACT ON MISSION OR CAPABILITY IF CHANGE NOT FUNDED AND REMARKS

15. COORD/ACTION OFFICERS	TYPE OR PRINT NAME	PHONE NO.	OFFICE SYMBOL	SIGNATURE
ORIGINATOR OF THIS FORM				
SYSTEM HARDWARE ENGINEER				
COMPUTER PROGRAMS ENGINEER				
DATA/DOCUMENTATION				
USER				

16. SUPPORT FACILITY CHIEF (Print or type Name)	17. CPCSB SECRETARY (Print or type Name)

18. SUB BOARD CHAIRMAN SIGNATURE AND TYPED NAME	☐ APPROVED ☐ DISAPPROVED ☐ DEFERRED ☐ ALC CCB DATE ________
19. ALC CCB CHAIRMAN SIGNATURE AND TYPED NAME (When applicable)	☐ APPROVED ☐ DISAPPROVED ☐ DEFERRED ☐ AFLC CCB DATE ________
20. HQ AFLC CCB CHAIRMAN SIGNATURE AND TYPED NAME (When applicable)	☐ APPROVED ☐ DISAPPROVED ☐ DEFERRED DATE ________

AFLC Form 75 (Page 2 of 2)

Appendix B: Detailed Description of AFLC Form 75

The first few entries on the Form 75 are administrative, with the identifying of the date the form was prepared, the system designator, and the weapon system or ground system being supported (3:59). Block number three indicates whether the change is Class I (usually CPCI changes to OFPs) or Class II (typically does not affect program logic) which can be verified by reference to MIL-STD-483 (24:122). Administrative functions are continued in block number four requiring the name of the "agency or specific command organization originating change" (3:59).

The information collected in block five is the start of information for the serious analysis of the software change. To correctly complete this form, each line must have at least one box marked. In line 'A', the scope of the change is expressed as either a software only change or the result of a hardware change. If this software change is the result of a hardware change then an AFLC Form 18, Configuration Control Board Item Record, (see Appendix C), must be included as part of the modification package (3:45). This information is required to help determine if the software is being modified in conjunction with a hardware change or because of a already existing phenomena.

The type of software changed is indicated on line 'B' as being used for an aircrew training device (ATD), automatic test equipment (ATE), communications-electronics

(C-E), electronic warfare (EW), operational flight program (OFP), commercial off- the-shelf software (COTS/FSG-70), or other (3:53-6). If the 'other' category is selected, an explanatory note is included in block ten. When a hardware modification affects more than one system, then a separate Form 75 is completed for each area (3:59). This information on system maintenance can indicate where more design work needs to be implemented to prevent later, more expensive, maintenance actions.

The indication of who is to complete the change in software is shown in line 'C', with choices that include organic, contractor, or other. Again, if 'other' is chosen, a specific explanation is recorded in block 10. The choice of who is to complete the maintenance action can also be a combination of organic and contractor support, which would be indicated by marking both boxes (3:59). There are several uses for this information including the tracking of the number of systems that are unsupportable by organic means, the types of systems that are contracted out for maintenance, and highlight areas that show dangerous trends in lack of wartime capability.

The last element in block five is line 'D', and is used to show whether this is a version change, a revision change, or a new CPCI. The amount of actual code, form, and function that is changed indicates whether it is a version change or a revision change (3:43).

One of the most important areas of concern in software maintenance is making sure that everyone who is affected by a change is aware of the change. This is the purpose of block number six, as it provides a list of the agencies that will be involved in the change (24:122). This is especially true when the software change affects more than one type of aircraft. This was illustrated on a recent software change to the AN/ALR-69, which affected F-16s, A-10s, F-4s, C-130s, and MH-53s, in such diverse places as PACAF, AAC, AFRES, ATC, TAC, SAC, MAC, and ANGSC (15:2).

The purpose of block seven is to guarantee that all parties are referencing the same software change. The first entry is the local control number, which is assigned by the Computer Program Configuration Sub-Board (CPCSB). This number is an alphanumeric with seven positions, with the first position a 'S', the second is an ALC identifier, i.e. 2 - San Antonio, 3 - Sacramento, 4 - Oklahoma City, 5 - Ogden, and 6 - Warner Robins. The third position is the last digit of the calendar year the Form 75 was submitted. The fourth through sixth positions are part of a consecutive numbering system for the division's annual software change traffic. The last character indicates the responsible division in the ALC for this CPCI (3:45).

The second number in this block is the CPIN, and as was previously described, remains with the particular CPCI at all times. If more than one CPIN is affected then a notation is made to "See Block 10" for more information and the complete list of affected CPINs (3:59). The third number identified in this block is the number of the deficiency report (DR) that identified the need for a change. If more than one DR is being corrected in the same change, list them in block ten (3:59). Careful consideration must be given to the number of deficiencies that are corrected at any one time. If too many deficiencies are included in a block change then the complexity of the change may become difficult to handle. On the other hand, if too many small changes are continually produced, then the users may become disgruntled with the instability of the software (12:10).

The fourth number is the identification number of the Engineering Change Proposal (ECP) that is responsible for the actual implementation of this change (24:123).

The fifth number in this block is the number of the Material Improvement Project (MIP) which is associated with this change, if more than one MIP, list them in block ten (3:59).

The sixth number in this block is the Time Compliance Technical Order (TCTO) associated with this change. An essential step of any software change is to document the change. Bad or incorrect documentation is worse than no documentation and most documentation is incorrect (17:173).

The Contract Engineering Project (CEP) number is the next item in this block and is used when the change is software only and performed by other than organic efforts (24:123).

The last item in this block is the hardware Modification (MOD) number and is used only if the software change is a result of a hardware change. Block five 'A', "Result of Hardware Mod", must be checked in order to have a MOD number (24:123).

Block eight indicates the relative priority of the change that is requested. This priority is based upon the urgency that each user attributes to the underlying discrepancies. The overall priority assigned to block changes are based upon several factors including required resources, individual discrepancy priority, current corrective operations and previously scheduled deadlines (3:45). Each ALC is responsible for the establishment of guidelines for the handling of high priority workloads (3:14). The personnel at WR-ALC have devised a plan that will allow the complete processing of an emergency change within 72 hours of receiving the operational change request. This is accomplished by working around-the-clock, and performing many actions simultaneously. Urgent requests are handled in 72 hours also, but only in the course of the normal duty day. This means that urgent requests are completed in nine working days, instead of three calendar days (24:52). On routine block changes, the scheduled release date is entered in this block, and if anything other than routine is checked, an explanation is required in block ten (24:123).

In block nine the specific CPCI/CPINs that are related to, affected by, or associated with, the change/development under consideration are listed (3:59). This provides the reviewers of the change request a better understanding of the possible repercussions of the change.

Block ten is used to describe the overall direction of the software change. This is done by first explaining the purpose of the change, in an unclassified manner. If classified information is necessary, provide it in an approved OPSEC/COMSEC manner (3:59). The next step in block ten, is to describe the procedure that will be carried out in the change including, "specific details of change describing impact on program, functions to be modified, data inputs or outputs changed, arithmetic and logic functions affected" (3:59). As previously mentioned, the identification of hardware and software documentation that will require updating is important and is the next item included in block ten. Also in this section is the identification of the hardware, software, and personnel actions that will occur as a result of this change. Highlights of the coordination, documentation, and full implementation requirements of the technical and managerial aspects of this change are also included in this section (3:59). Finally, a verbal schedule for the implementation of this change is outlined, and if an interim solution is required, a proposed work-around is also provided in this section (3:59).

In block eleven a graphical schedule for the completion of the change is given. Depending on the length of time required, the timeline can be in hourly, weekly, monthly, or quarterly increments. Some milestones included are formal approval, test/validation & verification, distribution, and receipt by the user. Included in the distribution area are milestones for the preparation of documentation for engineering and technical requirements. Interim actions are on this schedule, when required. A separate milestone chart may be attached if necessary (3:59-60).

Block twelve is designed to identify the man-hours, funding codes, and dollars associated with the change. The resource requirements are listed for both contractor and organic support of engineering, testing, distributing, and documenting the change. The cost of materiel such as disks, tapes, chips and other program media will also be included in this estimate. An allocation is also made in this section for the administrative support associated with the change. When this software change is the result of a hardware modification, the cost of the hardware is also included (3:60).

Block thirteen is a checklist to answer three basic questions about the proposed software change. The first question wants to know if the cost estimates in block twelve were budgeted for in the Program Objective Memorandum and the Operations Operating Budget. If the answer for this question is no, then an explanation is included in block fourteen. The second question wants to know if the hardware modification has been funded. The third question wants to know if the software change is on OFP, EW, or C-E, do the computer programs for the ATE or ATD have to be changed. If the answer to this last question is yes, then notify the appropriate Item or System Manager (3:60).

Block fourteen is used to explain the impact of not approving the change, and how it would affect the fulfillment of the mission. Noted in this block also are the explanations of how the cost data was computed. In addition comments are given in this block on how ATE costs are affected by changes in EW and OFP (3:60).

Block fifteen is used to confirm coordination with all essential parties, and obtain their written approval before the change occurs. If any questions develop at a later time, this coordination provides a starting point to discover the reasoning behind the original approval (3:60).

Block sixteen is used to obtain the written approval of the Support Facility Chief. This approval verifies that he has reviewed the Form 75 and sees no insurmountable problems in scheduling, budgeting, or allocating resources for the change (3:60).

Block seventeen is for the signature of the Computer Program Configuration Sub-board secretary. This demonstrates that he has reviewed the Form 75 for correctness and finds no major discrepancies at this time (3:60).

Blocks eighteen, nineteen, and twenty are for the signature of the appropriate chairman. This signature can mean approval, disapproval, or deferment. If the action is deferred, further guidance must be given (3:60).

Appendix C: AFLC Form 18

CCB MODIFICATION REQUIREMENTS AND APPROVAL DOCUMENT

*** FOR OFFICIAL USE ONLY ***

1. Mod Number: ____ 2. Mod Manager ALC: OO 3. Document Prep Date: ________ 4. Document Production Date: ________
5. Mod Title: ____________ 6a. System/Equip: ________ b. Asn: ____
7. Mod Class: ___ 8. Using Command: __ 9. Agencies Involved: a. USAF( ) b. SAP( ) c. Other( ) 10. Level of Competition: ___
11. Kit Installation Level: ____________ 12a. Installation Hours per Unit: ____
b. Total Installation Hours: ____

13a. Modification Mgr: ________ at ________ AV: 458 - Ext: ____
b. Project Officer: ________ at ________ AV: 458 - Ext: ____

14. Where Engineering Source Obtained: a. AFLC ( ) b. AFSC ( ) c. Contractor ( ) d. Other ( )

15. Description /Justification (narrative) : Description:

: Justification:

16. Status of Modification (narrative) :

17. Related Document Numbers:
a. MIP or SON Number: ________ b. PMD Number: ________ c. PAD Number: ________
d. TCTO number (s): ________ ________ ________ ________
e. MSTG Number: ________

18. Kit Qtys Req/Appl: a. Sys/Equip b. Spares c. [illegible] d. Trainers e. Simulators f. Other g. Total Kits

19. Action. a. New Proposal ( ) b. Add'l Requirement ( ) c. Cost Increase ( ) d. Schedule Slippage ( ) e. Cancellation ( ) f. Revalidation ( ) g. Summary ( )

20. Cost and schedule estimates herein must be revalidated if modification is not approved by this date: ________ (D/M/Y)

(Date derived from 1990 BES) Section 1 of 5
(Last entered/changed by : ______) Page 1 AFLC Form 18, Apr 89

*** FOR OFFICIAL USE ONLY ***

AFLC Form 18 (Page 1 of 5)

*** FOR OFFICIAL USE ONLY ***

21. Financial Plan, Mod # ____________ RPAC ______________

Preparation Date: ______

	FPY: 86		PPY: 87		PY: 88		CY: 89		AY: 90		BY: 91		BY+1: 92	
	-Qty-	-Cost-	-Qty-	-Cost-	-Qty-	-Cost-	-Qty-	-Cost-	-Qty-	-Cost-	-Qty-	-Cost-	Qty-	-Cost--
a. A&B KIT ENGINEERING														
b. ENGR CHANGE ORDERS														
c. DATA/MANUALS														
d. 'A' KITS/MATERIAL														
e. 'A' KITS NONRECUR														
f. 'B' KITS/MATERIAL														
g. 'B' KITS NONRECUR														
h. SPARE MOD KITS														
i. PECULIAR SUPT EQUIP														
j. SIMULATORS														
k. TRAINERS														
l. TOOLING														
m. OTHER														
n. BP11/21/8X COST SUBTOTAL	___	___	___	___	___	___	___	___	___	___	___	___	___	___
o. KIT PROOF LABOR														
p. OTHER LABOR (Installation)														
q. WRSK/BLSS SPARES INV														
r. WRSK/BLSS SPARES EXP														
s. INITL POS SPARES INV														
t. INITL POS SPARES EXP														
u. SOFTWARE (583)														
v. RDT&E (3600)														
w. COMMON SUPT EQUIPMENT														
x. OTHER														
y. TOTAL ALL COSTS	___	___	___	___	___	___	___	___	___	___	___	___	___	___

	BY+2: 93		BY+3: 94		BY+4: 95		BY+5: 96		OUT YR: ___				TOTAL	
	-Qty-	-Cost-	-Qty-	-Cost-	-Qty-	-Cost-	-Qty-	-Cost-	-Qty-	-Cost-			--Qty-	--Cost--
a. A&B KIT ENGINEERING														
b. ENGR CHANGE ORDERS														
c. DATA/MANUALS														
d. 'A' KITS/MATERIAL														
e. 'A' KITS NONRECUR														
f. 'B' KITS/MATERIAL														
g. 'B' KITS NONRECUR														
h. SPARE MOD KITS														
i. PECULIAR SUPT EQUIP														
j. SIMULATORS														
k. TRAINERS														
l. TOOL[illegible]														
m. OTHER														
n. BP11/21/8X COST SUBTOTAL	___	___	___	___	___	___	___	___	___	___	___	___	___	___
o. KIT PROOF LABOR														
P. OTHER LABOR (INSTALLATION)														
q. WRSK/BLSS SPARES INV														
r. WRSK/BLSS SPARES EXP														
s. INITL POS SPARES INV														
t. INITL POS SPARES EXP														
u. SOFTWARE (583)														
v. RDT&E (3600)														
w. COMMON SUPT EQUIPMENT														
x. OTHER														
y. TOTAL ALL COSTS	___	___	___	___	___	___	___	___	___	___	___	___	___	___

CCB Modification
Requirements & Approval

(All costs inflated, in $Millions)

Page 2

Section 2 of 5
AFLC Form 18, Apr 89

*** FOR OFFICIAL USE ONLY ***

AFLC Form 18 (Page 2 of 5)

*** FOR OFFICIAL USE ONLY ***

22. Applicable Lead Times in months, Mod# ______, BPAC ____: Preparation Date: ______

a. Initial Administrative: ___ + Production: b. ___/c. ___/d. ___ = e. Total: ____

f. Follow-on Administrative: ___ + g. Production: ____ = h. Total: ____ Dock Time: ____

23. Milestones:

FY ___ O : N : D : J : F : M : A : M : J : J : A : S | FY ___ O : N : D : J : F : M : A : M : J : J : A : S

a. ECP Date (________)
b. CCB Date (________)
c. Plan PR Dt (________)
d. Contract Award Date (s)
e. Trl Inst Dt (________)
f. Kit Prf Dt (________)
g. Kit Delivery Date (s)
h. Kit Instl Qtys, Depot
i. Kit Instl Qtys,, DFT

FY ___ O : N : D : J : F : M : A : M : J : J : A : S | FY ___ O : N : D : J : F : M : A : M : J : J : A : S

a. ECP Date (________)
b. CCB Date (________)
c. Plan PR Dt (________)
d. Contract Award Date (s)
e. Trl Inst Dt (________)
f. Kit Prf Dt (________)
g. Kit Delivery Date (s)
h. Kit Instl Qtys, Depot
i. Kit Instl Qtys, DFT

FY ___ O : N : D : J : F : M : A : M : J : J : A : S | FY ___ O : N : D : J : F : M : A : M : J : J : A : S

a. ECP Date (________)
b. CCB Date (________)
c. Plan PR Dt (________)
d. Contract Award Date (s)
e. Trl Inst Dt (________)
f. Kit Prf Dt (________)
g. Kit Delivery Date (s)
h. Kit Instl Qtys, Depot
i. Kit Instl Qtys, DFT

FY ___ O : N : D : J : F : M : A : M : J : J : A : S | FY ___ O : N : D : J : F : M : A : M : J : J : A : S

a. ECP Date (________)
b. CCB Date (________)
c. Plan PR Dt (________)
d. Contract Award Date (s)
e. Trl Inst Dt (________)
f. Kit Prf Dt (________)
g. Kit Delivery Date (s)
h. Kit Instl Qtys, Depot
i. Kit Instl Qtys, DFT

24. Infla Cost Summaries by Budget Program Activity:

a. 3100	b. 1200	c. 3500	d. 1600	e. Expense	f. DPEM	g. 583	h. 3600	i. Other
____	____	____	____	____	____	____	____	____

25. Average Raw Kit Expense (per assembly) = ____________ (All costs in $Millions)

CCB Modification Requirements & Approval

FOR OFFICIAL USE ONLY

Section 3 of 5

Page 3

AFLC Form 18, Apr 89

*** FOR OFFICIAL USE ONLY ***

AFLC Form 18 (Page 3 of 5)

*** FOR OFFICIAL USE ONLY ***

MOD# ______ PREPARATION DATE: ______
BPAC ______

CCB COORDINATION PAGE

26. Technical risk
   a. HIGH ( ) (explain in comments)
   b. LOW ( )

27. Cost risk
   a. HIGH ( ) (explain in comments)
   b. LOW ( )

28. a. Mod specs available? ( )
   b. Specs revision required? ( )
   c. Date available: (______)

29. Have alternate means of satisfying this requirement been investigated?
   a. YES ( )
   b. NO ( )

30. a. Mod approved for production/procurement cut-on? ( )
   b. Cut-in Item Serial Number: (________)
   c. Forecast Item Delivery Date: (________)

31. Coordination

	NAME	Office	DATE
a. SPM/IM			
b. LSSPO			
c. LSSM			
d. Sim OOALC			
e. TRNR OOALC			
f. TRNR ATC			
g. MAD			
h. ASD/AFALC/AXAC			
i. ______			
j. ______			
k. ______			
l. ______			
m. ______			
n. ______			
o. ______			

32. Responsible SPM
   a. Approved ______
   b. Disapproved ___
   c. Signature:
   ____
   d. Date: ______________

33. ALC CCB Chairperson
   a. Approved _____
   b. Disapproved _____
   c. Approved for referl to AFLC ___
   d. Approved for referl to ALC ___
   e. Signature:
   ____
   f. Date: ______________

34. AFLC CCB Chairperson
   a. Approved ___
   b. Disapprved ___
   c. Approved for referl to USAF ___
   d. Signature:
   ____
   e. Date: ______________

35. USAF
   a. Approved ___
   b. Disapproved ___
   c. Deffered ___ until:
   Date: __________
   d. Signature:
   ____
   e. Date ______________

36. Comments

CCB MODIFICATION
Requirements & Approval

Page 4

ATCH 1
Section 4 of 5
AFLC Form 18, Apr 89

*** FOR OFFICIAL USE ONLY ***

AFLC Form 18 (Page 4 of 5)

*** FOR OFFICIAL USE ONLY ***

MDS*
RPAC

PREPARATION DATE:

CCB REVIEW PAGE

37. MODIFICATION REVIEW CHECKLIST

	YES	NO
a INTEGRATED LOGISTIC SUPPORT PLAN (ILSP) REQUIRED (AFRS 800-8/57-4)	( )	( )
b REVISION OF DESIGN HANDBOOKS OR ITEM SPECS	( )	( )
c FLIGHT MANUALS AFFECTED	( )	( )
d SPEEDLINE OPERATIONS AFFECTED	( )	( )
e SYSTEM/EQUIPMENT CALIBRATION AFFECTED	( )	( )
f IMPACT ON THE ENVIRONMENT (AFRS 19-1 & 19-2)	( )	( )
g HUMAN FACTOR ENGINEERING COORDINATION REQUIRED (AFSCR/AFLCR 80-17)	( )	( )
h COST/SCHEDULE CONTROL SYSTEM CRITERIA (AFR 800-6)	( )	( )
i OSHA STANDARDS CONSIDERED (AFR 127-12 AFLC SUP 1)	( )	( )
j NON-NUCLEAR MUNITIONS SAFETY INVOLVED (AFLC SUP 1 TO AFR 127-16)	( )	( )
k NUCLEAR SAFETY CERTIFICATION REQUIRED (AFRS 122-3 & -9)	( )	( )
l FIBER OPTICS TECHNOLOGY CONSIDERED	( )	( )
m ACCOUNTING REVIEW OF HIGH COST ESTIMATES ACCOMPLISHED	( )	( )
n AIRCRAFT/STORES COMPATABILITY CERTIFICATION REQUIRED (AFLCR/AFSCR 80-28)	( )	( )
o MOD AFFECTS SURVIVABILITY/VULNERABILITY (AFLC SUP 1 TO AFR 80-38)	( )	( )
p PRELIMINARY HAZARD ANALYSIS REQUIRED (AFR 800-16 & MIL-STD-882B)	( )	( )
q SUBSYSTEM HAZARD ANALYSIS TO INCLUDE SYSTEM INTERFACE REQUIRED (MIL-STD-882B, TASK 203)	( )	( )
r SYSTEM HAZARD ANALYSIS REQUIRED (MIL-STD-882B, TASK 204)	( )	( )
s OPERATING & SUPPORT HAZARD ANALYSIS REQUIRED (MIL-STD-882B, TASK 205)	( )	( )
t RISK ASSESSMENT PROCESS REQUIRED (CLASS IVA ONLY) (AFR 57-4, TABLE 1, NOTE 5)	( )	( )
u CORROSION POTENTIAL ITEMS AFFECTED	( )	( )
v IMPACT OF MOD ON RELIABILITY/MAINTAINABILITY CONSIDERED	( )	( )
w NON-DESTRUCTIVE INSPECTION CONSIDERED	( )	( )
x ENVIRONMENTAL STRESS SCREENING CONSIDERED (MIL-STD-785B, TASK 301)	( )	( )
y ANY POTENTIAL GROUP B TECHNICAL INTERFACE IMPACTS HAVE BEEN EVALUATED BY AFFECTED SPOs AND SPMs	( )	( )
z TEST AND EVALUATION REQUIREMENTS CONSIDERED (AFR 80-14)	( )	( )

38. Review Remarks

CCB MODIFICATION
Requirements & Approval

Page 5

ATCH 2
SECTION 5 OF 5
AFLC Form 18, Apr 89

*** FOR OFFICIAL USE ONLY ***

AFLC Form 18 (Page 5 of 5)

Terminology

AFWAL	Air Force Wright Aeronautical Laboratories
AFLC	Air Force Logistics Command
ALC	Air Logistics Center
ASSCM	Avionics Software Support Cost Model
CE	Communications Equipment
COCOMO	Constructive Cost Model
DASO	Defense Studies and Analysis Office
EW	Electronic Warfare
IEEE	Institute of Electrical/Electronics Engineers
NBS	National Bureau of Standards
OC-ALC	Oklahoma City, Air Logistics Center
OO-ALC	Ogden, Air Logistics Center
OFP	Operational Flight Program
O&M	Operation and Maintenance
OSD	Office of the Secretary of Defense
PDSS	Post Deployment Software Support
SA-ALC	San Antonio, Air Logistics Center
SM-ALC	Sacramento, Air Logistics Center
SCC	Software Control Center
SSC	Software Support Center
SYSCON	Systems Consultants, Inc.
WR-ALC	Warner Robins, Air Logistics Center

Bibliography

1. Air Force Institute of Technology. Military Logistics. School of Systems and Logistics. Wright-Patterson AFB OH, May 1988.

2. Air Force Logistics Command. Handbook for Engineers on Funding Resources for Reliability and Maintainability Programs. DCS/Material Management. Wright-Patterson AFB OH, May 1987.

3. -----. Management and Support Procedures for Computer Resources Used in Defense Systems. AFLCR 800-21. Wright-Patterson AFB OH, 21 January 1983.

4. Air Force Management Engineering Agency. Peacetime Military Man-Hour Availability Factor (MAF) Update. Studies and Analysis Division. Randolph AFB TX, October 1987.

5. Appleton, Gary R., Deputy Chief, Weapon Systems Support Division, Dir/Maintenance & Supply, HQ USAF. Personal Correspondence. Washington, 4 November 1988.

6. Boehm, Barry W., Software Engineering Economics. Englewood Cliffs NJ: Prentice-Hall Inc., 1981.

7. Brooks, Frederick P., Jr. The Mythical Man-Month. Reading MA: Addison-Wesley Publishing Company, 1981.

8. Canan, James W., "The Software Crisis," Air Force Magazine, 69: 46-52 (May 1986).

9. Daft, Richard L. and Richard M. Steers. Organizations: A Micro/Macro Approach. Glenview IL: Scott, Foresman and Company, 1986.

10. Davis, Malcom R. and others. Aquisition and Support of Embedded Computer System Software, September 1981. Contract F49620-82-C-0018. Santa Monica CA: Rand Corporation (AD-A121420).

11. Defense Analysis and Studies Office. Software Maintenance in DoD. Falls Church VA: 12 December 1988.

12. Department of the Air Force. Lifecycle Management of Computer Resources in Systems. AFR 800-14. Washington: HQ USAF, 29 September 1986.

13. Ferens, Daniel V., Mission Critical Computer Software Support Management. Text written for Course SYS 202, Mission Critical Computer Software Support Management. School of Systems and Logistics, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, May 1987.

14. Grove, H. Mark "DoD Policy for Acquisition of Embedded Computer Resources," Concepts, The Journal of Defense Systems Acquisition Management, 5: 9-36 (Autumn 1982).

15. Kalejta, Len. MCCR Focal Point. Personal Correspondence. Warner-Robins Air Logistics Center, May 1989.

16. Kitfield, James. "Is Software DoD's Achilles' Heel?," Military Forum, 5: 28-35 (July 1989).

17. Martin, James and Carma McClure. Software Maintenance: The Problem and Its Solutions. Englewood Cliffs NJ: Prentice-Hall Inc., 1983.

18. National Bureau of Standards. Guidance on Software Maintenance. Special Publication 500-106. Washington: Government Printing Office, December 1983.

19. -----. Software Maintenance Management. Special Publication 500-129. Washington: Government Printing Office, October 1985.

20. Parikh, Girish. How to Measure Programmer Productivity. Chicago: Shetal Enterprises, 1981.

21. Swanson, E. B. "The Dimensions of Maintenance," Proceedings IEEE/ACM 2nd International Conference on Software Engineering. 492-497. Washington: Computer Society Press of the IEEE, 1978.

22. SYSCON. Avionics Software Support Cost Model: Final Report, September 1980 - November 1982. Contract F33615-80-C-1157. Washington: SYSCON Corporation, 1 February 1983 (AD-A128523).

23. Waina, Richard B. and others. Predictive Software Cost Model Study: Final Technical Report, 2 April 1979 - 2 June 1980. Contract F33615-79-C-1734. Canoga Park CA: Hughes Aircraft Company, June 1980 (AD-A088476).

24. Warner Robins Air Logistics Center. Configuration Management of EW System Software. MMR Operating Instruction 55-1. DCS/Material Management. Robins AFB GA, 1 September 1986.

25. Webb-Johnson, Lord. "Talk Around Town," _Look, 19_: 16 (4 October 1955).

26. _Webster's New World Dictionary_. College Edition. New York: The World Publishing Company, 1966.

Vita

Captain Mark A. Collette [redacted] in Kankakee, Illinois. [redacted] in 1975 and subsequently enlisted in the Marine Corps as an Avionics Technician. Upon completion of his tour he attended Eastern Oregon State College and graduated with a Bachelor of Science in Business Economics in 1984. He was then commissioned through the Air Force OTS program on 21 August 1985. From August 1985 to May 1988, he was assigned to the 56th Tactical Training Wing, MacDill AFB, Florida. During this time, Captain Collette served as the Assistant Chief of the Resource Plans Division. In June 1988, he entered the School of Systems and Logistics, Air Force Institute of Technology.

[redacted]

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION
UNCLASSIFIED

1b. RESTRICTIVE MARKINGS

2a. SECURITY CLASSIFICATION AUTHORITY

3. DISTRIBUTION/AVAILABILITY OF REPORT
Approved for public release; distribution unlimited

2b. DECLASSIFICATION/DOWNGRADING SCHEDULE

4. PERFORMING ORGANIZATION REPORT NUMBER(S)
AFIT/GLM/LSY/89S-10

5. MONITORING ORGANIZATION REPORT NUMBER(S)

6a. NAME OF PERFORMING ORGANIZATION
School of Systems and Logistics

6b. OFFICE SYMBOL *(If applicable)*
AFIT/ GLM

7a. NAME OF MONITORING ORGANIZATION

6c. ADDRESS *(City, State, and ZIP Code)*
Air Force Institute of Technology (AU)
Wright-Patterson AFB, Ohio 45433-6583

7b. ADDRESS *(City, State, and ZIP Code)*

8a. NAME OF FUNDING/SPONSORING ORGANIZATION

8b. OFFICE SYMBOL *(If applicable)*

9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER

8c. ADDRESS *(City, State, and ZIP Code)*

10. SOURCE OF FUNDING NUMBERS

PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	WORK UNIT ACCESSION NO.

11. TITLE *(Include Security Classification)*
AN INQUIRY INTO THE COSTS OF POST DEPLOYMENT SOFTWARE SUPPORT (PDSS) (UNCLASSIFIED)

12. PERSONAL AUTHOR(S)
Mark A. Collette, B.S., Capt, USAF

13a. TYPE OF REPORT
MS Thesis

13b. TIME COVERED
FROM ______ TO ______

14. DATE OF REPORT *(Year, Month, Day)*
1989 September

15. PAGE COUNT
68

16. SUPPLEMENTARY NOTATION

17. COSATI CODES

FIELD	GROUP	SUB-GROUP
12	05	

18. SUBJECT TERMS *(Continue on reverse if necessary and identify by block number)*
Software Support, Acquisition, Embedded Computer Systems

19. ABSTRACT *(Continue on reverse if necessary and identify by block number)*

Thesis Advisor: Daniel V. Ferens
Associate Professor of Systems Management
Department of System Acquisition Management

Approved for public release: IAW AFR 190-1.

Larry W. Emmelhainz

LARRY W. EMMELHAINZ, Lt Col, USAF 14 Oct 89
Director of Research and Consultation
Air Force Institute of Technology (AU)
Wright-Patterson AFB OH 45433-6583

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT
☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS

21. ABSTRACT SECURITY CLASSIFICATION
UNCLASSIFIED

22a. NAME OF RESPONSIBLE INDIVIDUAL
Daniel V. Ferens, Professor

22b. TELEPHONE *(Include Area Code)*
(513) 255-4845

22c. OFFICE SYMBOL
AFIT/LSY

DD Form 1473, JUN 86 *Previous editions are obsolete.* SECURITY CLASSIFICATION OF THIS PAGE

UNCLASSIFIED

Block 19.

Abstract

The increasing cost of software maintenance is taking a larger share of the military budget each year. As new weapon systems are acquired, these support costs must be accurately estimated and budgeted for in the total cost of acquisition. In particular, the accurate tracking of Post Deployment Software Support (PDSS) costs is critical to the overall estimation process, yet it is one of the most difficult areas to document.

This study had three objectives achieve to be considered successful:

(1) Determine why PDSS is so costly and difficult to estimate.

(2) Determine what factors make PDSS costs fluctuate.

(3) Develop possible changes to the PDSS tracking system to improve availability of information.

The conclusions that were reached by this thesis indicate that the redesign of the AFLC Form 75 is a partial solution to the software support tracking system. The use of the Form 75 as a funding document would appear to be a necessary step in the process toward a more complete cost estimating system.